UNIVERSITÄT ZU LÜBECK
INSTITUTE OF SOFTWARE ENGINEERING
AND PROGRAMMING LANGUAGES

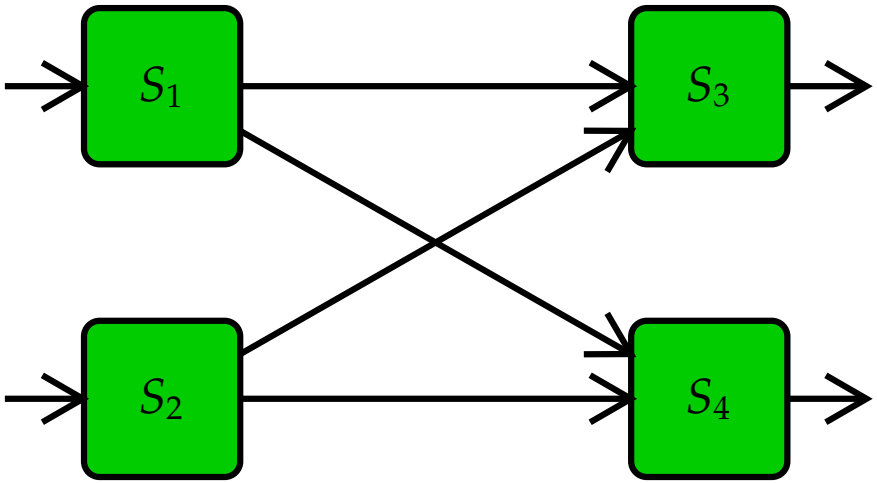
isp

Runtime Verification with Predictive Semantics

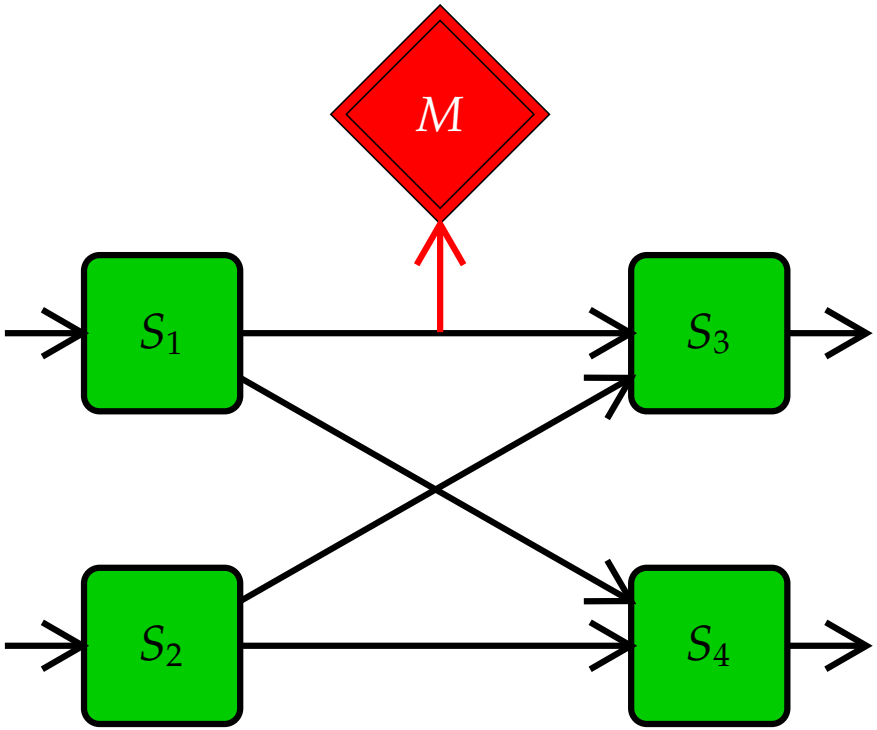
Xian Zhang Martin Leucker Wei Dong

Norfolk, April 2012

Runtime Verification (RV)

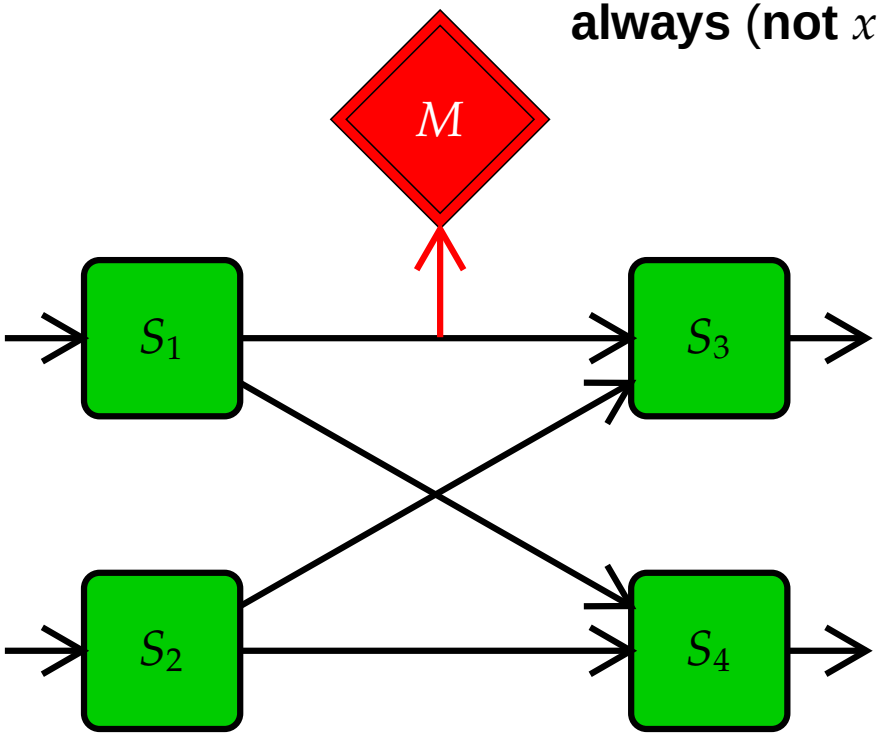


Runtime Verification (RV)



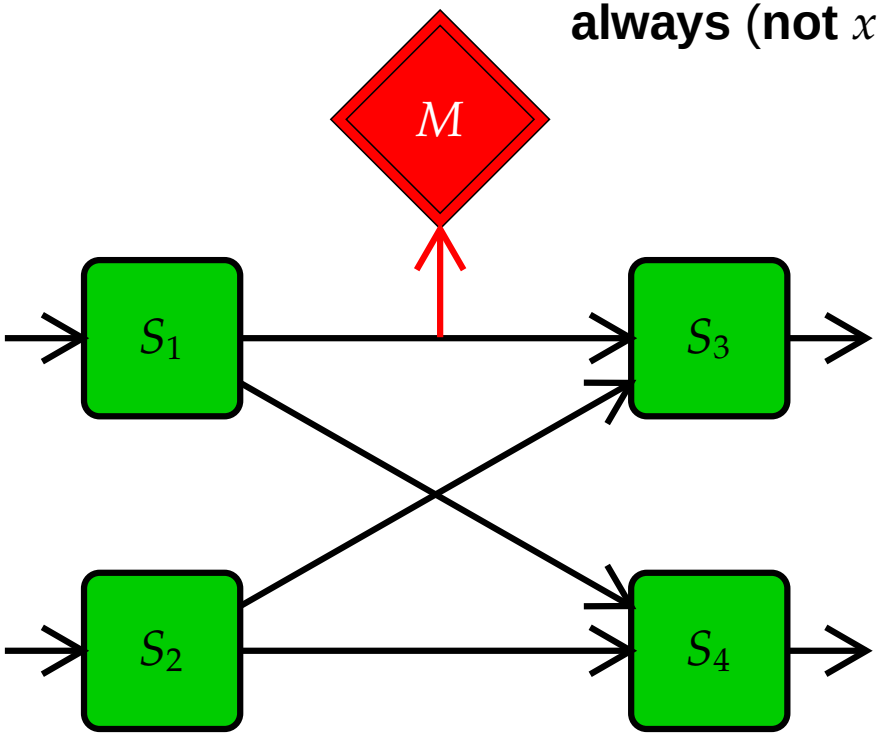
Runtime Verification (RV)

always (not $x > 0$ **implies next** $x > 0$**)**



Runtime Verification (RV)

always (**not** $x > 0$ **implies next** $x > 0$)

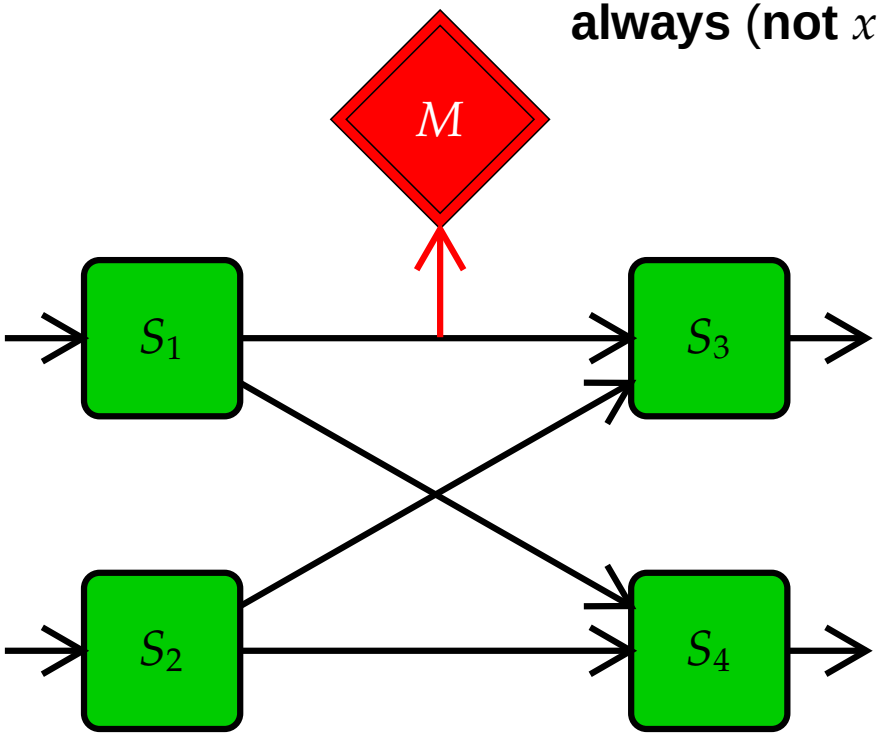


Characterisation

- Verifies (partially) correctness properties based on actual executions

Runtime Verification (RV)

always (not $x > 0$ **implies next** $x > 0$**)**

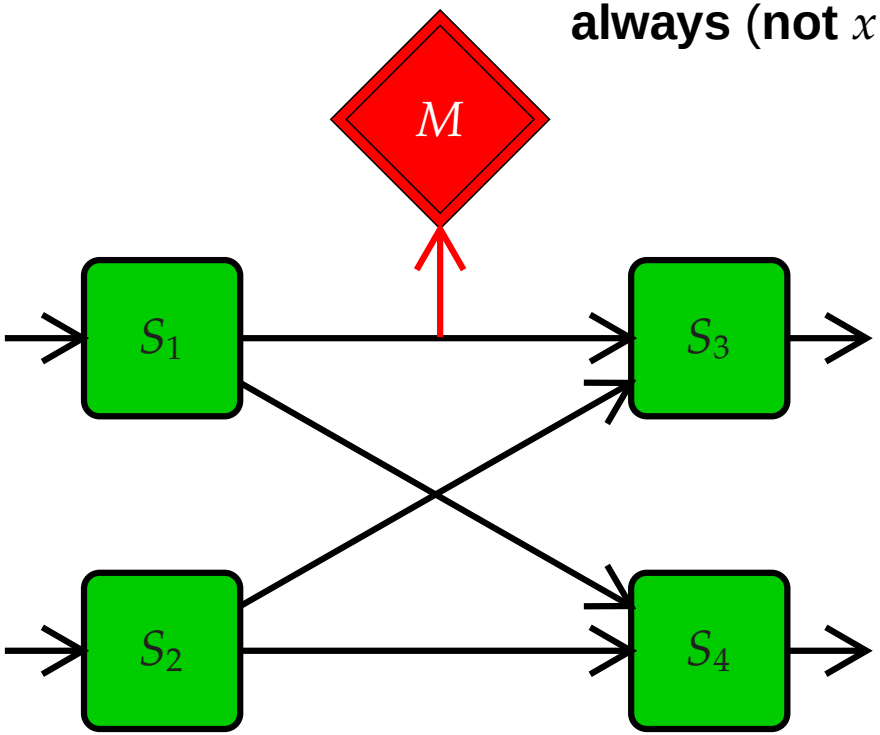


Characterisation

- Verifies (partially) correctness properties based on actual executions
- Simple verification technique

Runtime Verification (RV)

always (not $x > 0$ **implies next** $x > 0$**)**

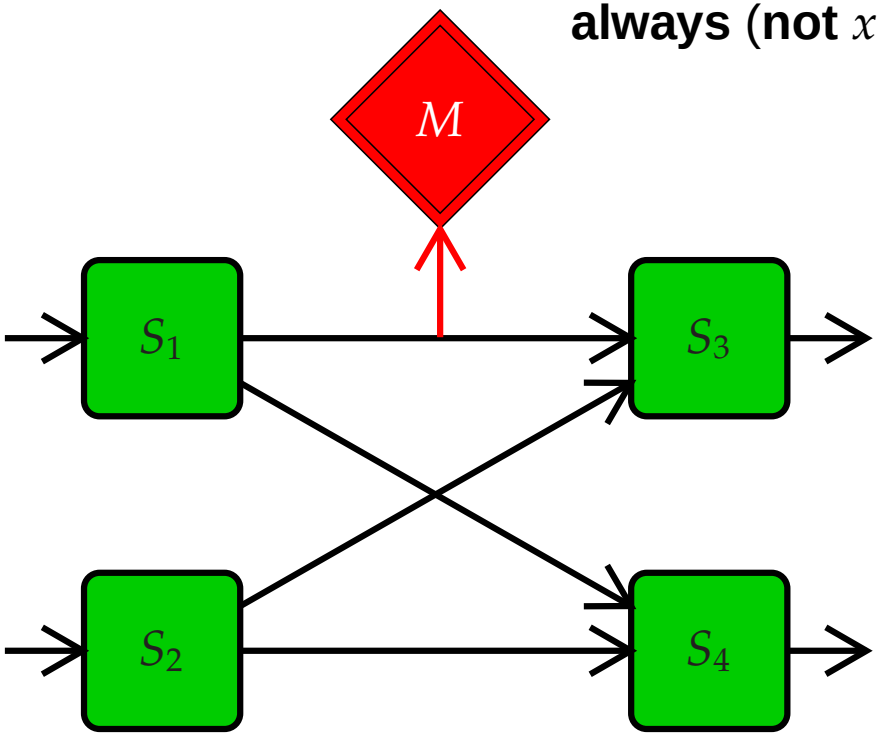


Characterisation

- Verifies (partially) correctness properties based on actual executions
- Simple verification technique
- Complementing

Runtime Verification (RV)

always (not $x > 0$ **implies next** $x > 0$**)**

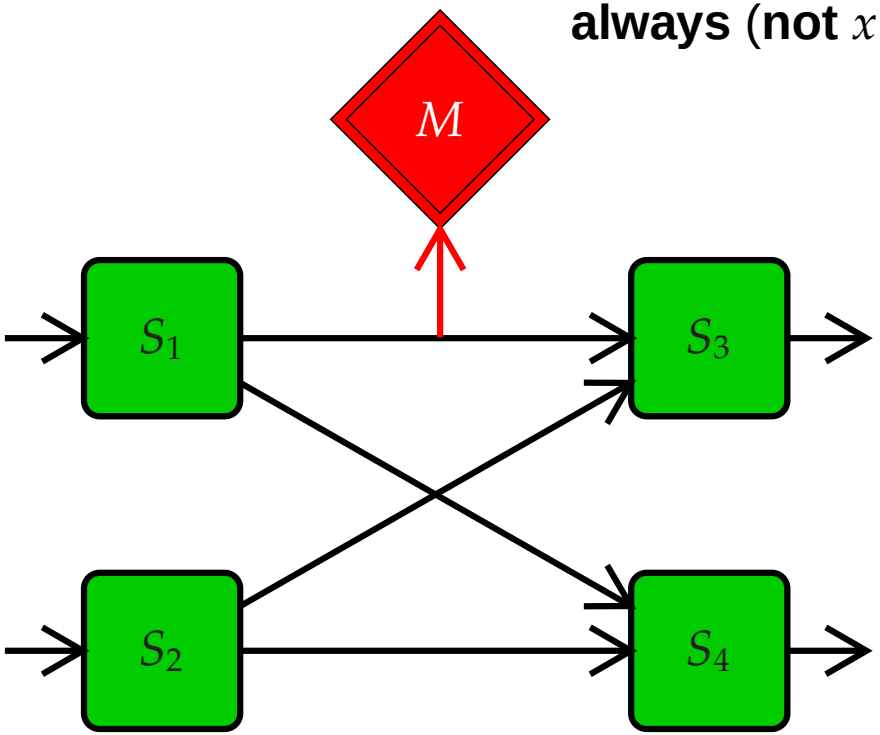


Characterisation

- Verifies (partially) correctness properties based on actual executions
- Simple verification technique
- Complementing
  - Model Checking

Runtime Verification (RV)

always (not $x > 0$ **implies next** $x > 0$**)**

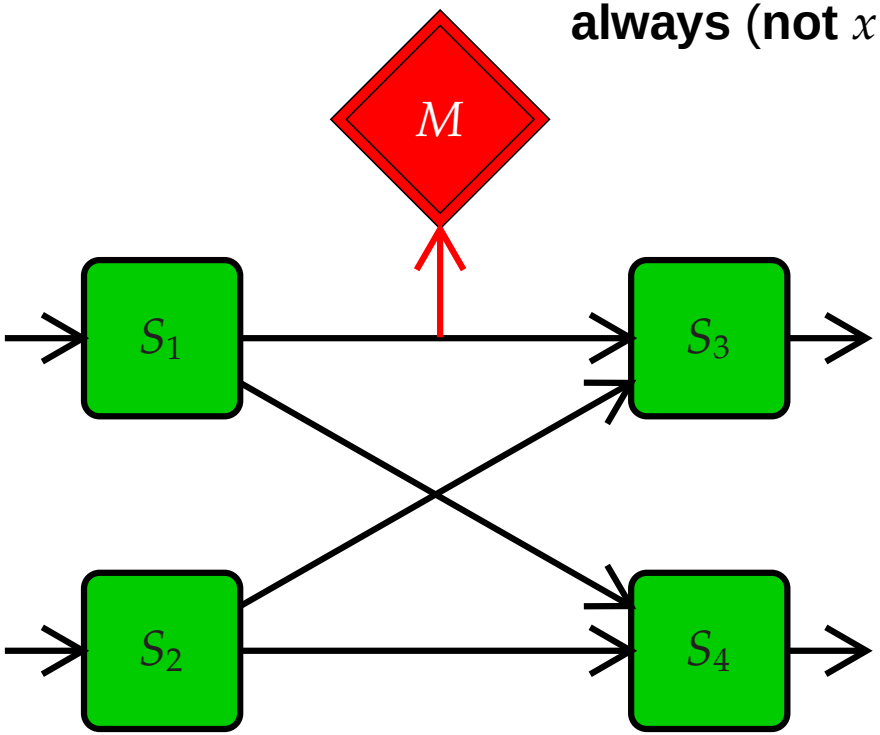


Characterisation

- Verifies (partially) correctness properties based on actual executions
- Simple verification technique
- Complementing
  - Model Checking
  - Testing

Runtime Verification (RV)

always (not $x > 0$ **implies next** $x > 0$**)**



Characterisation

- Verifies (partially) correctness properties based on actual executions
- Simple verification technique
- Complementing
  - Model Checking
  - Testing
- Formal: $w \in \mathcal{L}(\varphi)$

Outline

Presentation outline

Runtime Verification for LTL

Observing executions/runs

Runtime Verification for LTL

Observing executions/runs

Idea

Specify correctness properties in LTL

Runtime Verification for LTL

Definition (Syntax of LTL formulae)

Let p be an atomic proposition from a finite set of atomic propositions AP. The set of LTL formulae, denoted with LTL, is inductively defined by the following grammar:

$$\begin{aligned} \varphi \quad ::= \quad & \mathit{true} \mid p \mid \varphi \vee \varphi \mid \varphi \, U \, \varphi \mid X\varphi \mid \\ & \mathit{false} \mid \neg p \mid \varphi \wedge \varphi \mid \varphi \, R \, \varphi \mid \bar{X}\varphi \mid \\ & \neg\varphi \end{aligned}$$

Truth Domains

Lattice

- A lattice is a partially ordered set $(\mathcal{L}, \sqsubseteq)$ where for each $x, y \in \mathcal{L}$, there exists
  1. a unique greatest lower bound (glb), which is called the meet of x and y, and is denoted with $x \sqcap y$, and
  2. a unique least upper bound (lub), which is called the join of x and y, and is denoted with $x \sqcup y$.
- A lattice is called finite iff $\mathcal{L}$ is finite.
- Every finite lattice has a well-defined unique least element, called bottom, denoted with $\bot$,
- and analogously a greatest element, called top, denoted with $\top$.

Truth Domains (cont.)

Lattice (cont.)

- A lattice is distributive, iff $x \sqcap (y \sqcup z) = (x \sqcap y) \sqcup (x \sqcap z)$, and, dually, $x \sqcup (y \sqcap z) = (x \sqcup y) \sqcap (x \sqcup z)$.
- In a de Morgan lattice, every element x has a unique dual element $\overline{x}$, such that $\overline{\overline{x}} = x$ and $x \sqsubseteq y$ implies $\overline{y} \sqsubseteq \overline{x}$.

Definition (Truth domain)

We call $\mathcal{L}$ a truth domain, if it is a finite de Morgan lattice.

LTL's semantics using truth domains

Definition (LTL semantics (common part))

Semantics of LTL formulae over a finite or infinite word $w = a_0 a_1 \ldots \in \Sigma^\infty$

Boolean constants

$$[w \models \mathit{true}]_{\mathfrak{L}} = \top \qquad [w \models \mathit{false}]_{\mathfrak{L}} = \bot$$

Boolean combinations

$$[w \models \neg\varphi]_{\mathfrak{L}} = \overline{[w \models \varphi]_{\mathfrak{L}}}$$
$$[w \models \varphi \vee \psi]_{\mathfrak{L}} = [w \models \varphi]_{\mathfrak{L}} \sqcup [w \models \psi]_{\mathfrak{L}}$$
$$[w \models \varphi \wedge \psi]_{\mathfrak{L}} = [w \models \varphi]_{\mathfrak{L}} \sqcap [w \models \psi]_{\mathfrak{L}}$$

atomic propositions

$$[w \models p]_{\mathfrak{L}} = \begin{cases} \top & \text{if } p \in a_0 \\ \bot & \text{if } p \notin a_0 \end{cases} \qquad [w \models \neg p]_{\mathfrak{L}} = \begin{cases} \top & \text{if } p \notin a_0 \\ \bot & \text{if } p \in a_0 \end{cases}$$

next X/weak next X TBD

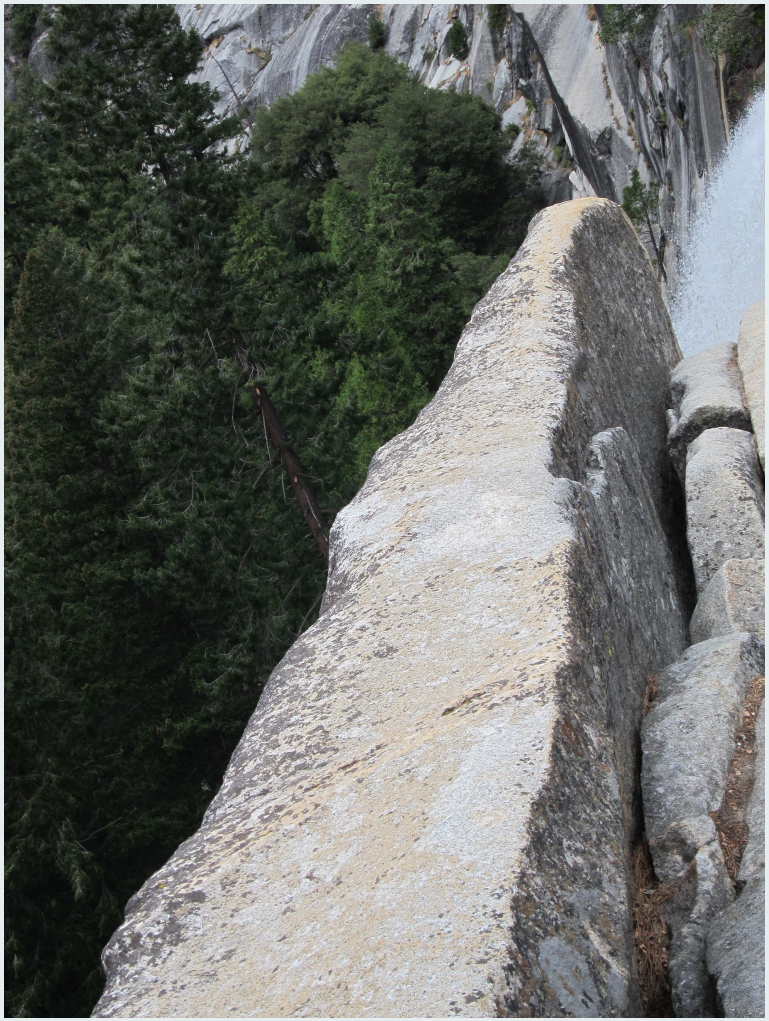
until/release

$$[w \models \varphi \; U \; \psi]_{\mathfrak{L}} = \begin{cases} \top & \text{there is a } k,\ 0 \le k < |w| : [w^k \models \psi]_{\mathfrak{L}} = \top \text{ and} \\ & \text{for all } l \text{ with } 0 \le l < k : [w^l \models \varphi] = \top \\ \mathit{TBD} & \text{else} \end{cases}$$

$$\varphi \; R \; \psi \equiv \neg(\neg\varphi \; U \; \neg\psi)$$

LTL on finite words

Application area: Specify properties of finite word



LTL on finite words

Definition (FLTL)

Semantics of FLTL formulae over a word $u = a_0 \dots a_{n-1} \in \Sigma^*$

next

$$[u \models X\varphi]_F = \begin{cases} [u^1 \models \varphi]_F & \text{if } u^1 \neq \epsilon \\ \bot & \text{otherwise} \end{cases}$$

weak next

$$[u \models \bar{X}\varphi]_F = \begin{cases} [u^1 \models \varphi]_F & \text{if } u^1 \neq \epsilon \\ \top & \text{otherwise} \end{cases}$$

LTL on finite, but not completed words

Application area: Specify properties of finite but expanding word

LTL on finite, but not completed words

Be Impartial!

- go for a final verdict ($\top$ or $\bot$) only if you really know

LTL on finite, but not complete words

Impartiality implies multiple values

Every two-valued logic is not impartial.

Definition (FLTL)

Semantics of FLTL formulae over a word $u = a_0 \dots a_{n-1} \in \Sigma^*$

next

$$[u \models X\varphi]_F = \begin{cases} [u^1 \models \varphi]_F & \text{if } u^1 \neq \epsilon \\ \bot^p & \text{otherwise} \end{cases}$$

weak next

$$[u \models \bar{X}\varphi]_F = \begin{cases} [u^1 \models \varphi]_F & \text{if } u^1 \neq \epsilon \\ \top^p & \text{otherwise} \end{cases}$$

Anticipatory Semantics

Consider possible extensions of the non-completed word

LTL for RV [BLS@FSTTCS'06]

Basic idea

- LTL over infinite words is commonly used for specifying correctness properties
- finite words in RV:
  prefixes of infinite, so-far unknown words
- re-use existing semantics

LTL for RV [BLS@FSTTCS'06]

Basic idea

- LTL over infinite words is commonly used for specifying correctness properties
- finite words in RV:
  prefixes of infinite, so-far unknown words
- re-use existing semantics

3-valued semantics for LTL over finite words

$$[u \models \varphi] = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^\omega : u\sigma \models \varphi \\ \bot & \text{if } \forall \sigma \in \Sigma^\omega : u\sigma \not\models \varphi \\ ? & \text{else} \end{cases}$$

Impartial Anticipation

Impartial

- Stay with $\top$ and $\bot$

Impartial Anticipation

Impartial

- Stay with $\top$ and $\bot$

Anticipatory

- Go for $\top$ or $\bot$
- Consider $XXXfalse$

$$\epsilon \quad \models \quad XXXfalse$$

Impartial Anticipation

Impartial

- Stay with $\top$ and $\bot$

Anticipatory

- Go for $\top$ or $\bot$
- Consider $XXXfalse$

$$\epsilon \models XXXfalse$$
$$a \models XXfalse$$

Impartial Anticipation

Impartial

- Stay with $\top$ and $\bot$

Anticipatory

- Go for $\top$ or $\bot$
- Consider $XXXfalse$

$$\epsilon \models XXXfalse$$
$$a \models XXfalse$$
$$aa \models Xfalse$$

Impartial Anticipation

Impartial

- Stay with $\top$ and $\bot$

Anticipatory

- Go for $\top$ or $\bot$
- Consider $XXXfalse$

$$\begin{aligned} \epsilon &\models XXXfalse \\ a &\models XXfalse \\ aa &\models Xfalse \\ aaa &\models false \end{aligned}$$

$$[\epsilon \models XXXfalse] = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^\omega : \epsilon\sigma \models XXXfalse \\ \bot & \text{if } \forall \sigma \in \Sigma^\omega : \epsilon\sigma \not\models XXXfalse \\ ? & \text{else} \end{cases}$$

Presentation outline

Predictive Semantics

Consider the program to monitor

LTL with Predictive Semantics

Basic idea

- finite words in RV:
  prefixes of infinite, so-far unknown words

LTL with Predictive Semantics

Basic idea

- finite words in RV:
  prefixes of infinite, so-far unknown words of our program

LTL with Predictive Semantics

Basic idea

- finite words in RV:
  prefixes of infinite, so-far unknown words of our program

A first predictive semantics for LTL over finite words

$$[u \models \varphi] = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^\omega \text{ with } u\sigma \in \mathcal{P} : u\sigma \models \varphi \\ \bot & \text{if } \forall \sigma \in \Sigma^\omega \text{ with } u\sigma \in \mathcal{P} : u\sigma \not\models \varphi \\ ? & \text{else} \end{cases}$$

Too much. . .

Answers model checking question!

$$[\epsilon \models \varphi] = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^{\omega} \text{ with } \sigma \in \mathcal{P} : \epsilon\sigma \models \varphi \\ \bot & \text{if } \forall \sigma \in \Sigma^{\omega} \text{ with } \sigma \in \mathcal{P} : \epsilon\sigma \not\models \varphi \\ ? & \text{else} \end{cases}$$

More reasonable. . .

Use abstraction

- Use overabstraction of $\hat{\mathcal{P}}$
- with $L(\mathcal{P}) \subseteq L(\hat{\mathcal{P}}) \subseteq \Sigma^\omega$

More reasonable. . .

Use abstraction

- Use overabstraction of $\hat{\mathcal{P}}$
- with $L(\mathcal{P}) \subseteq L(\hat{\mathcal{P}}) \subseteq \Sigma^\omega$

A general predictive semantics

$$[u \models \varphi] = \begin{cases} \top & \text{if } \forall \sigma \in \Sigma^\omega \text{ with } u\sigma \in \hat{\mathcal{P}} : u\sigma \models \varphi \\ \bot & \text{if } \forall \sigma \in \Sigma^\omega \text{ with } u\sigma \in \hat{\mathcal{P}} : u\sigma \not\models \varphi \\ ? & \text{else} \end{cases}$$

But . . .

How to get $\hat{\mathcal{P}}$?

- here, use simple analysis of $\mathcal{P}$
- find for $\mathcal{P}$ sequential executions of actions over φ's alphabet
- obtain finite set R of

But . . .

How to get $\hat{\mathcal{P}}$?

- here, use simple analysis of $\mathcal{P}$
- find for $\mathcal{P}$ sequential executions of actions over φ's alphabet
- obtain finite set R of

Predictive semantics

$$[w \models^R \varphi] = \begin{cases} \top & \text{if } \forall v \in R\, \forall \sigma \in \Sigma^\omega : uv\sigma \models \varphi \\ \bot & \text{if } \forall v \in R\, \forall \sigma \in \Sigma^\omega : uv\sigma \not\models \varphi \\ ? & \text{else} \end{cases}$$

Well . . .

Really. . .

- find sequences av in $\mathcal{P}$
- send av to monitor rather than only a

Presentation outline

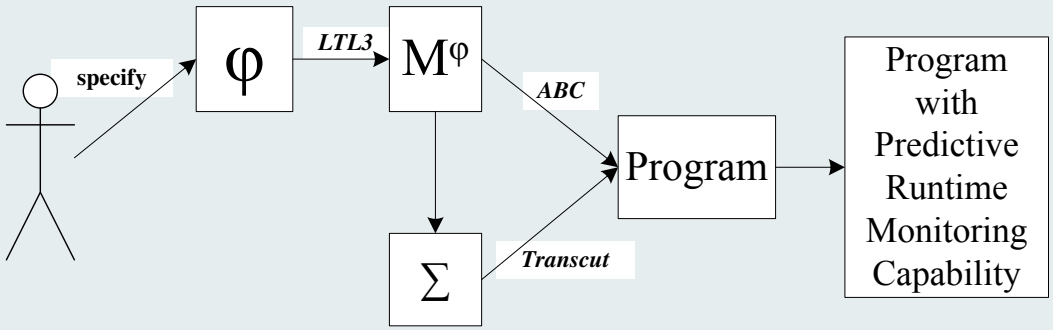
Implementation

Instrumentation

- use of AspectJ (*abc*) to obtain events from program
- analysis of strong regions, CFG and PDG to find sequential executions
- use *Transcut* to inject predictive words

Implementation

A typical usage scenario of our prototype tool



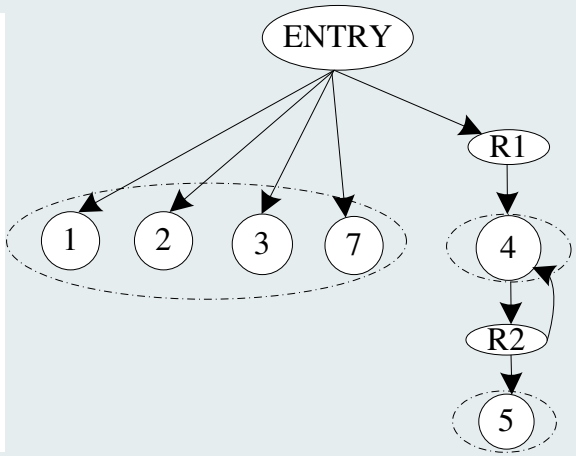
Implementation

Property

$G(\texttt{create} \rightarrow G(\texttt{update} \rightarrow \neg F(\texttt{next})))$

Strong Regions and PDG

```
1 Vector v = new Vector(col)
2 Iterator it = v.iterator( );
3 v.add(object);
4 for(each element in col){
5     process each element ;
6 }
7 it.next();
```



Experimental Results

Setup

- used Dacapo benchmarks

Results

	antlr	eclipse	fop	hsqldb	bloat	lucene
class number	224	344	967	385	263	311
method number	2972	3978	6889	5859	3986	3013
predictable shadow ratio	0% (0/23)	7.92% (53/391)	24.65% (83/288)	28.23% (45/124)	17.06% (608/1495)	25% (61/224)
predictable region ratio	0% (0/23)	3.33% (22/360)	7.86% (24/229)	11.83% (14/93)	7.88% (204/1091)	7.3% (15/178)

Presentation outline

That's it!

Thanks! - Questions?